MARATONA DE PROGRAMAÇÃO

InterFatecs

IV MARATONA DE PROGRAMAÇÃO INTERFATECs 2015

www.fateccruzeiro.edu.br/interfatecs

1ª. Fase - 09 de maio de 2015

Caderno de Problemas

Este caderno contém 10 problemas – A a J, com páginas numeradas de 01 a 21.
Verifique se o caderno está completo.

Informações Gerais

A) Sobre o arquivo de solução e a submissão

1. O arquivo de solução deve ter o mesmo nome que o especificado no enunciado (logo após o título);
2. Confirme se você escolheu a linguagem correta e está com o nome de arquivo correto antes de submeter a sua solução.

B) Sobre a entrada

1. A entrada de seu programa deve ser lida da entrada padrão (não use interface gráfica);
2. A entrada é composta por vários casos de teste, especificados no enunciado;
3. Em alguns problemas, o final da entrada coincide com o final do arquivo. Em outros, o final é especificado por alguma combinação de caracteres ou dígitos.

C) Sobre a saída

1. A saída de seu programa deve ser escrita na saída padrão;
2. Não exiba qualquer mensagem não especificada no enunciado.

Promoção:

Fatec Cruzeiro Prof. Waldomiro May

Centro Paula Souza

Governo do Estado São Paulo – Secretaria de Desenvolvimento Econômico, Ciência, Tecnologia e Inovação

Apoio:

UNISAL Centro Universitário Salesiano de São Paulo

EMC²

Cengage Learning

mf Soluções em Tecnologia

Camisetas evidence – Qualidade e conforto para o seu dia a dia

Problema A

Código de barras

Arquivo fonte: `barcode.{c | cpp | java}`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Os códigos de barra são um grande avanço tecnológico e organizacional para a vida moderna, pois permitem uma agilidade muito grande na identificação dos produtos a serem manipulados nas várias operações do dia a dia de uma empresa. Existem algumas variações na forma de codificação, mas a ideia é basicamente a mesma: um prefixo numérico chamado *System Number*, o *Manufacturer Code*, o *Product Code* e o *Check Digit*. No Brasil e em muitos países do mundo é utilizada a versão EAN-13 dos códigos de barra, que será apresentada a seguir.

Fig. 1: Exemplo de um código de barras no formato EAN-13

Na figura 1 temos um exemplo de código de barras no formato explorado neste problema. O código total é composto por 13 dígitos numéricos, o que explica em parte o nome EAN-13. Os dois primeiros dígitos são os chamados números do sistema (no caso, 76); os próximos 5 dígitos formam o chamado *Manufacturer Code* (que no exemplo é 12345); os 5 dígitos seguintes compõem o chamado *Product Code* (67890 em nosso exemplo) e o dígito restante é o chamado *Check Digit* (o dígito verificador, que é calculado com base nos 12 primeiros dígitos e que em nosso exemplo é 0). Toda etiqueta de código de barras possui o código representado em duas formas: a forma numérica decimal, que aparece na parte inferior da figura, e a forma gráfica em colunas, que é aquele conjunto de linhas verticais característico desse tipo de código visando a leitura rápida por meio de dispositivos óticos especiais. O leitor ótico reconhece as colunas escuras da etiqueta como sendo o valor 1 e as colunas brancas como sendo o valor 0. Repare na figura que o primeiro dígito não é expresso por colunas, ao contrário dos dígitos restantes. Temos inicialmente um conjunto de três colunas que correspondem a um delimitador padrão obrigatório chamado *Left Guard*: uma coluna escura, uma coluna clara, uma coluna escura, que é entendida pelo leitor como sendo 101. Em seguida temos seis dígitos expressos por meio de arranjos de 7 colunas cada, que correspondem ao segundo dígito do número do sistema e aos 5 dígitos do *Manufacturer Code*.

Depois existe um novo delimitador padrão obrigatório, chamado *Central Guard*, que é formado por uma coluna clara, uma escura, uma clara, uma escura e outra clara, que é reconhecida pelo leitor como sendo a sequência 01010. Então aparecem os 6 dígitos finais, cada um expresso por um arranjo de 7 colunas, que correspondem aos 5 dígitos do *Product Code* e ao dígito verificador. A codificação se encerra com o *Right Guard*, que é composto por uma coluna escura, uma clara e outra escura, formando a sequência 101.

Então, cada dígito numérico decimal do código corresponde a uma combinação de zeros e uns de comprimento 7. Para determinar qual a representação de zeros e uns de cada dígito, precisamos levar em conta uma série de fatores. O primeiro deles é em que lado do *Central Guard* está o dígito: se estiver antes desse delimitador, ou seja, do seu lado esquerdo, a representação tem um critério, se o dígito estiver depois do delimitador, ou seja, do seu lado direito, a representação tem outro critério. Para os dígitos do lado esquerdo, precisamos saber inicialmente o valor do primeiro dígito, aquele que é o primeiro número do sistema e que não aparece nas colunas. A Tabela 1 indica como o segundo, terceiro, quarto, quinto e sexto dígitos devem ser codificados em barras, de acordo com o valor desse primeiro dígito. Já para os dígitos do lado direito, a codificação é sempre a mesma, como mostra a Tabela 2.

Primeiro dígito	**Díg. 2**	**Díg. 3**	**Díg. 4**	**Díg. 5**	**Díg. 6**	**Díg. 7**
0	Ímpar	Ímpar	Ímpar	Ímpar	Ímpar	Ímpar
1	Ímpar	Ímpar	Par	Ímpar	Par	Par
2	Ímpar	Ímpar	Par	Par	Ímpar	Par
3	Ímpar	Ímpar	Par	Par	Par	Ímpar
4	Ímpar	Par	Ímpar	Ímpar	Par	Par
5	Ímpar	Par	Par	Ímpar	Ímpar	Par
6	Ímpar	Par	Par	Par	Ímpar	Ímpar
7	Ímpar	Par	Ímpar	Par	Ímpar	Par
8	Ímpar	Par	Ímpar	Par	Par	Ímpar
9	Ímpar	Par	Par	Ímpar	Par	Ímpar

Tabela 1: Mapa de paridade dos dígitos do lado esquerdo do código de barras EAN-13

Dígito	**Lado Esquerdo**		**Lado Direito**
	Paridade Ímpar	**Paridade Par**	
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000

9	0001011	0010111	1110100

Tabela 2: Representação em colunas dos dígitos decimais do código de barras

O código de exemplo 7612345678900 teria então a seguinte codificação em colunas: `101` (*Left Guard*) `0101111` (o '6' que corresponde ao primeiro conjunto de colunas) `0110011` (o '1' que corresponde ao segundo conjunto de colunas) `0010011` ('2') `0100001` ('3') `0100011` ('4') `0111001` ('5') `01010` (*Central Guard*) `1010000` ('6') `1000100` ('7') `1001000` ('8') `1110100` ('9') `1110010` ('0') `1110010` ('0') `101` (*Right Guard*). Juntando tudo temos:

```
10101011110110011001001101000010100011011100101010101000010001
00100100011101001110010111001010
```

Isso quer dizer que se um leitor ótico lesse a etiqueta da Figura 1, geraria exatamente essa sequência de zeros e uns.

O dígito verificador, por sua vez, é calculado de uma maneira muito simples: somam-se os dígitos decimais localizados nas posições ímpares do código. Em nosso exemplo, seria 7 + 1 + 3 +5 + 7 + 9 = 32. Depois geramos um segundo total, pela soma dos dígitos das posições pares (menos o último dígito, que é o que queremos calcular). Em nosso exemplo seria 6 + 2 +4 + 6+ 8 + 0 = 26. Adicionamos à primeira soma o triplo da segunda, o que em nosso exemplo corresponde a 32 + 3 * 26 = 32 + 78 = 110. O dígito verificador é o módulo da diferença entre esse número e o primeiro múltiplo de 10 maior ou igual a esse número. Em nosso exemplo produzimos o valor 110, que por coincidência, é também múltiplo de 10. Então o dígito verificador procurado é 110 – 110 = 0, ou seja, o *Check Digit* para o código do exemplo da Figura 1 é 0.

Sua tarefa neste problema é, dado um conjunto de leituras de código de barras realizada por um leitor ótico, verificar se o dígito verificador está correto ou não.

Entrada

A entrada é composta por vários casos de teste, cada um expresso por uma linha contendo uma sequência de 95 caracteres zeros e uns que corresponde à leitura do reconhecedor ótico. A entrada é sinalizada pelo final de arquivo.

Saída

Para cada caso de teste, imprima uma linha com a mensagem 'barcode incorreto: lido = *N* esperado = *X*', caso o dígito verificador esteja incorreto. Nesse caso *N* representa o dígito verificador lido na entrada e *X* o dígito verificador correto para aquele código de barras. A mensagem deverá estar totalmente em minúsculas. Caso o dígito verificador esteja correto, imprimir o número do sistema, o *Manufaturer Code*, o *Product Code* e o dígito verificador, separados por um hífen, como mostram os exemplos.

Exemplos

Entrada:

```
10101011110110011001001101000010100011011100101010101000010001001001
00011101001110010111001010 1
10101110110001001000100101100010100001000101101010111001011100101001
11010000101001110110110010 1
10100010110100111011001100100110111101001110101010110011011011001000
01010111001001110100010010 1
10100100110001101010011101001110100111000110101010111001011100
10100001010001001000100100111010 1
```

Saída:

```
76-12345-67890-0
97-88539-00535-2
59-01234-12345-7
barcode incorreto: lido = 5 esperado = 4
```

Dica de solução (problema A):

O problema requer que se valide o dígito verificador de cada código de barras lido na entrada de dados. Se o dígito lido estiver correto, imprime-se a sequência numérica correspondente àquele código de barras; se houver divergência entre o dígito verificador lido e o dígito calculado com base na fórmula apresentada no enunciado, o programa de imprimir a mensagem indicativa dos dois valores divergentes.

O primeiro passo é decodificar os dígitos decimais a partir da representação binária lida na entrada, o que é feito com base nas especificações das tabelas 1 e 2. O conjunto dos dígitos decimais de 2 a 7 indicam na Tabela 1 qual o valor do primeiro dígito decimal, que não está contido na representação binária.

Uma vez produzidos os dígitos decimais referentes ao código, basta aplicar a fórmula contida na parte final do enunciado e obter o Check Digit esperado, que será então comparado com o último dígito decimal da entrada.

Problema B

Museu

Arquivo fonte: `museu.{c | cpp | java}`

Autor: Emanuel Mineda Carneiro (FATEC-SJC)

Um museu recebeu um conjunto de obras de arte formadas por azulejos quadrados. Todos os azulejos possuem o mesmo tamanho e são totalmente coloridos nas cores branca ou preta.

O diretor do museu decidiu classificar as obras pela quantidade de figuras contida em cada uma. Uma figura é composta por azulejos vizinhos de cor preta. Dois azulejos são considerados vizinhos se um lado de um se encontra totalmente unido a um lado do outro. Desta forma, um azulejo pode possuir, no máximo, quatro vizinhos: um acima, um abaixo, um à direita e outro à esquerda.

A Tabela 1 apresenta um exemplo de obra de arte formada por 10 colunas de azulejos, com altura de 8 azulejos. Esta obra de arte apresenta 4 figuras, compostas por azulejos vizinhos de cor preta.

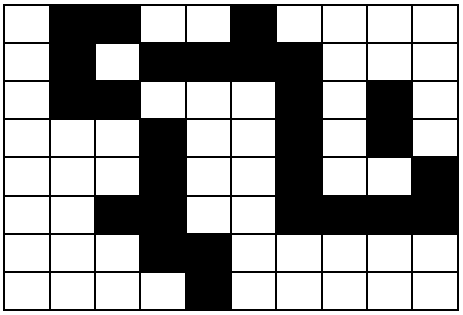


Tabela 1: Exemplo de obra de arte.

Você foi contratado para desenvolver um programa capaz de contar a quantidade de figuras contidas em cada obra de arte.

Entrada

A entrada é composta por vários casos de teste. A primeira linha de cada caso de teste contém a quantidade de azulejos na horizontal H ($0 < H < 15$) e na vertical V ($0 < V < 15$). As linhas seguintes contém a cor dos azulejos (0 para branco e 1 para preto), separados por um espaço em branco. Os casos de teste terminam com o final do arquivo (EOF).

Saída

Para cada caso de teste, deve ser impressa a quantidade de figuras encontradas na obra. Cada linha deve possuir o resultado de um único caso de teste.

Exemplos

Entrada:	**Saída:**
6 5 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 7 3 1 0 0 1 1 1 1 1 1 0 0 0 1 0 0 1 1 0 0 1 0	3 2

Dica de solução (problema B):

Sendo cada obra de arte uma matriz composta por 0s e 1s, espera-se que o programa percorra toda a matriz e, toda vez que encontrar um valor um 1, incrementar o contador de figuras e substituir todas as posições correspondentes à figura encontrada por valores diferentes de 1 (0, por exemplo).

A substituição pode ser realizada por um algoritmo recursivo:

- Parâmetros: posição (linha e coluna) a avaliar.

- Retorno: nenhum.

- Condição de parada: posição inválida ou conteúdo da posição diferente de 1.

- Ação: substitui o conteúdo da posição por 0 e chama o algoritmo recursivo para as 4 posições adjacentes/vizinhas (acima, abaixa, esquerda e direita).

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema C

Quadtree

Arquivo fonte: `quadtree.{c | cpp | java}`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Existem diversas formas práticas de compactação de informação, que fazem o seu trabalho por meio da eliminação de redundâncias presentes no contexto que se está analisando. Uma forma bastante interessante é por meio de estruturas denominadas Quadtree, que permitem uma decomposição recursiva do espaço. Uma quadtree é uma árvore onde todos os nós são folhas ou então possuem grau igual a 4. No caso mais simples, podemos assumir que as regiões de um espaço podem conter dois tipos de informação, que representaremos por zeros e uns neste problema. A ideia básica é subdividir o espaço em 4 partes, e atribuir a cada parte um valor 0 se aquela parte contiver apenas zeros, 1 se contiver apenas uns ou então 2, se a parte tiver tanto zeros como uns. Partes do tipo 2 são então subdivididas em 4 partes menores, e o processo continua até que não tenhamos quaisquer partes do tipo 2.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0
0	0	0	0	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	1

Figura 1a: Arranjo binário original

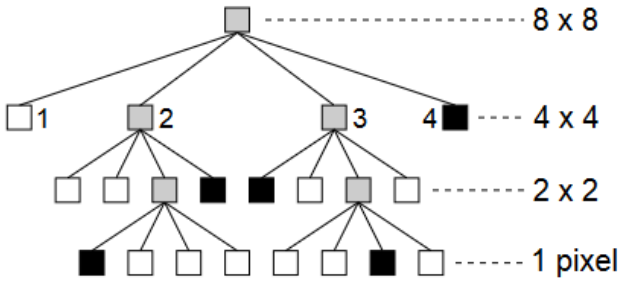


Figura 1b: Quadtree correspondente à Figura 1a

Na Figura 1 temos um arranjo binário disposto em um formato 8x8. A Figura 1a mostra o arranjo espacial na matriz quadrada subjacente, a Figura 1b mostra a quadtree correspondente. Na árvore a raiz corresponde à matriz 8x8, que contém tanto zeros como uns. Isso faz com que a raiz tenha valor 2 (indicado pela cor cinza na figura). Subdividimos então essa área não homogênea 8x8 em 4 pedaços 4x4: um no quadrante superior esquerdo, outro no superior direito, outro no quadrante inferior direito e outro no quadrante inferior esquerdo. O primeiro desses quadrantes tem apenas valores zero, então seu valor é zero (indicado pela cor branca do nó marcado com '1' na quadtree. O quadrante inferior esquerdo, por sua vez, possui apenas células (ou pixels) com valor um, então o quadrante inteiro tem valor 1 (indicado pela cor preta do nó marcado com '4' na quadtree). Os quadrantes restantes possuem tanto zeros como uns e estão marcados com cinza na figura, indicando que não são homogêneos e, por esse motivo, precisam ser subdivididos em arranjos 2x2. A quadtree da Figura 1b pode também ser expressa de forma textual como mostrado a seguir, onde o primeiro inteiro da primeira linha indica a quantidade de linhas do arranjo NxN e o segundo valor indica a cor do nó raiz da quadtree, seguindo-se então uma linha para cada nível da árvore, no sentido da esquerda para a direita:

```
8 2
0221
0021
1020
1000
0010
```

Sua tarefa neste problema será gerar a representação textual da quadtree correspondente a um arranjo binário NxN lido da entrada.

Entrada

Cada caso de teste é iniciado por um inteiro *N*, $4 \leq N \leq 512$, em que *N* é uma potência de 4 e indica a medida de cada lado do arranjo binário *N*x*N* a ser processado. Seguem-se então *N* linhas, cada uma contendo *N* valores *V* separados por um espaço em branco, onde o valor de *V* será sempre igual a zero ou a um. A entrada é finalizada com um valor N = 0, que não deverá ser processado.

Saída

Para cada caso de teste, imprima a representação textual da quadtree, conforme explicado anteriormente e ilustrado nos exemplos.

Exemplos

Entrada:	**Saída:**
8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 4 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0	8 2 0221 0021 1020 1000 0010 4 2 2222 0101 0101 0101 0101 8 2 0222 0011 1102 0210 1101 0111

Dica de solução (problema C):

O problema consiste em analisar um arranjo quadrado e imprimir os elementos da árvore correspondente. Para um quadrado verificamos se sua composição é homogênea (apenas zeros ou apenas uns). Caso seja, não há desdobramento, e o valor para o quadrado será igual ao valor contido em seus elementos. Se houver tanto zeros como uns será necessário subdividi-lo em 4 partes iguais, referentes aos quadrantes superior esquerdo (o primeiro), superior direito (o segundo), inferior direito (o terceiro) e inferior esquerdo (o quarto), repetindo-se o procedimento para cada subdivisão.

A solução pode ser implementada de forma iterativa com o auxílio de uma fila, que armazena as coordenadas das subdivisões a serem processadas, com o processamento se iniciando com as coordenadas do arranjo binário original e se encerrando quando a fila ficar vazia.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

Calendário Maia

Arquivo fonte: `maia.{c | cpp | java}`

Autor: Julio Fernando Lieira (Fatec Lins)

Usando seus conhecimentos de astronomia e matemática, os Maias desenvolveram um dos sistemas de calendário mais precisos da história da humanidade. Este sistema de calendários se baseia em diferentes ciclos de variados tamanhos. Os três mais conhecidos são: o calendário Haab, o qual considera os 365 dias do calendário solar e é usado como calendário civil; o Tzolk'in, é o calendário sagrado com ciclos de 260 dias baseado nos nove ciclos da Lua e no período de gestação humana; e o calendário de Contagem Longa, usado para registrar eventos míticos e históricos.

O Calendário de Contagem Longa é um sistema composto por uma hierarquia de ciclos de tempo, similar ao calendário Gregoriano que conta dias, meses, anos, séculos e milênios. Entretanto, como na matemática Maia, o calendário de Contagem Longa utiliza a base 20 na contagem dos ciclos, com exceção do terceiro ciclo, o qual utiliza 18x20, resultando em um ciclo de 360 dias, que é mais próximo do ciclo solar de 365 dias. A tabela 1 mostra estes ciclos.

Utilizando este sistema de contagem vigesimal modificado, este calendário conta os dias decorridos desde a data zero correspondente a 11 de agosto de 3113 a.C. (antes de Cristo) no calendário gregoriano. Embora seja composto por 9 ciclos, os achados arqueológicos mostram que os maias só utilizavam os 5 primeiros ciclos para registrar datas do passado, presente e futuro. Por exemplo, a data 20/10/1991 é escrita em Contagem Longa como sendo:

12.18.18.9.7

baktun.katun.tun.uinal.kin

```
12x144.000=1.728.000 dias
18x7.200=     129.600 dias
18x360=         6.480 dias
 9x20=            180 dias
 7x1=               7 dias
-------------------------------------------
Total         1.864.267 dias
```

Ou seja, passaram-se 1.864.267 dias após a data zero (11/08/3113 a.C.).

Ciclo	Multiplicador	Total em dias
kin		1 dia
uinal	20 x kin	20 dias
tun	18 x uinal	360 dias
katun	20 x tun	7200 dias
Baktun	20 x katun	144000 dias
Pictum	20 x baktun	2880000 dias
calabtun	20 x calabtun	57600000 dias
kinchiltun	20 x calabtun	1152000000 dias
alautun	20 x kinchiltun	23040000000 dias

Sua tarefa neste problema será construir uma calculadora para converter uma data no formato Gregoriano (dia/mês/ano) para o formato do calendário maia.

Entrada

Cada linha da entrada é composta por uma data no formato D/M/A, onde D, M, A são números inteiros e correspondem a uma data gregoriana. Note que as datas anteriores ao nascimento de Cristo estarão representadas pelo valor negativo do Ano (A). Lembre-se que

um ano Gregoriano tem 365 dias, ou 366 em casos de anos bissextos onde o mês de fevereiro tem 29 dias. O ano é bissexto quando é divisível por 4, exceto para anos divisíveis por 100 que, para serem bissextos, devem ser divisíveis também por 400. Assim, o ano 2000 é bissexto (divisível por 4, 100 e 400), mas os anos 1700, 1800 e 1900 NÃO são bissextos. Considere que $1 \leq D \leq 31$; $1 \leq M \leq 12$, $-3113 \leq A \leq 4000$. Considere também que todas as datas serão iguais ou posteriores a 11/8/3113 a.C.

Saída

Para cada data gregoriana da entrada, imprima na saída a data equivalente no formato Contagem Longa do calendário maia, considerando somente os 5 ciclos na forma: baktun.katun.tun.uinal.kin

Exemplos

Entrada:	**Saída:**
11/8/-3113	0.0.0.0.0
11/12/-3113	0.0.0.6.2
13/11/-2720	0.19.18.17.15
16/2/-2325	1.19.18.17.15
3/6/-354	6.19.18.17.15
1/1/1	7.17.18.13.3
20/10/1991	12.18.18.9.7
21/12/2012	13.0.0.0.0
1/1/4000	18.0.15.17.7

Dica de solução (problema D):

Para converter uma data Gregoriana no Formato de Contagem Longa, é preciso contar quantos dias se passaram desde a data 11/08/3113 AC até a data que se quer converter. Por exemplo, para converter a data 20/10/1991, dia em que Ayrton Sena foi tricampeão da F-1:

Primeiro calcula-se quantos dias se passaram de 11/08/3113 AC até o final do ano de 3113 AC

Depois calcula-se a quantidade de dias entre 3112 AC até o ano anterior ao ano da data final (1990, neste exemplo). Para tanto, basta fazer um laço de repetição variando de -3112 ao ano final -1 (1990, neste exemplo) e ir somando 365 ou 366 em caso de ano bissexto.

Por último calcula-se os dias decorridos dentro do ano final (neste exemplo, de 01/01/1991 a 20/10/1991).

Depois de ter a soma total de dias que se passaram desde 11/08/3113 AC até a data que se deseja converter, basta fazer a conversão nos ciclos maia, dividindo pela quantidade de dias (divisão inteira) de cada ciclo. Em Linguagem C:

```
baktun = contdias/144000;

katun = (contdias%144000)/7200;

tun = ((contdias%144000)%7200)/360;

uinal = (((contdias%144000)%7200)%360)/20;

kin = (((contdias%144000)%7200)%360)%20;
```

Note acima que o operador / é a divisão inteira e o operador % é o resto da divisão inteira.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

Matriz esparsa?

Arquivo fonte: `matriz.{c | cpp | java}`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Neste problema você colocará à prova todo o seu grande conhecimento sobre álgebra linear, acumulado por todo o ensino médio, cursinho e faculdade. Estamos interessados em um tipo muito especial de matriz quadrada: todos os elementos de sua diagonal principal possuem o valor 2; as diagonais secundárias imediatamente acima e abaixo da diagonal principal possuem o valor -1; os elementos restantes da matriz possuem o valor zero. As figuras abaixo mostram exemplos desse tipo de matriz. Vamos chamar de 'esparsa' uma matriz que possua um valor zero na maior parte de seus elementos. Sua tarefa é, para um dado tamanho de matriz do tipo descrito anteriormente, determinar se a mesma é esparsa ou não.

2	-1	0
-1	2	-1
0	-1	2

N = 3

2	-1	0	0
-1	2	-1	0
0	-1	2	-1
0	0	-1	2

N = 4

Entrada

Inicialmente um valor *Q* é informado, indicando a quantidade de casos de teste a serem processados. Seguem-se *Q* linhas, cada uma contendo um inteiro *N* indicando a quantidade de linhas da matriz a ser considerada. O valor de *Q* será um inteiro positivo menor ou igual a 100, e o valor de *N* será um inteiro positivo maior ou igual a 2 e menor ou igual a 100000.

Saída

Para cada caso de teste, imprima a letra S (em maiúscula) se a matriz for esparsa ou N (também em maiúscula) caso a matriz não o seja. Em ambos os casos, imprimir após a letra um inteiro indicando a quantidade de elementos com zero na matriz.

Exemplos

Entrada:	**Saída:**
5 4 2 5 100 101	N 6 N 0 N 12 S 9702 S 9900

Dica de solução (problema E):

Um dos problemas fáceis da prova, requeria se determinasse, para uma matriz quadrada, se ela era esparsa ou não e quantos elementos contendo o valor zero ela possui.

Como o enunciado informava que todos os elementos da diagonal principal têm valor 2; as duas diagonais vizinhas à diagonal principal têm em seus elementos o valor -1, e os elementos restantes da matriz possuem valor zero, bastava fazer o seguinte cálculo, considerando que a matriz tem dimensões N x N:

Quantidade total de elementos da matriz (T): N * N

Quantidade de elementos na diagonal principal (D1): N

Quantidade de elementos em cada diagonal vizinha à principal (D2): N – 1

Quantidade de elementos não-zero na matriz (U): D1 + 2 * D2

Quantidade de elementos com zero na matriz (Q): T – U

Para determinar o valor a ser exibido, considerar que a matriz é esparsa de Q > N / 2.

Soluções por simulação, que criam e preenchem a matriz e depois fazem a contagem dos elementos com valores zero e com valores diferentes de zero são muito ineficientes quando o valor de *N* for grande. Essas soluções não seriam aceitas na prova, pois o tempo limite definido para o problema não seria suficiente para que o programa resolvesse o problema.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem F

Programming Languages Popularity

Source code: `popularity.{c | cpp | java}`

Author: Leandro Luque (Fatec Mogi das Cruzes)

Javinha is carrying out a research about the popularity of programming languages as homework for a Programming Language course at Fatec. In a web search, he found some statistics on programming languages usage, but none of them satisfied him.

As he is an active participant of a discussion forum on programming language techniques, where programmers from all around the world send their questions and opinions, he decided to use the forum post archive to estimate the popularity of programming languages. The outline of his idea is as follows: the popularity of a specific programming language is equal to the number of posts in which this language is cited at least once.

Despite this simple idea, some problems may arise when searching for a citation in a post. For 'C', a search in a post like "as he can write a computer program" may return two false positives: "… Can write … Computer …". Said that, Javinha decided to consider as a citation only the name of a language with a space before and after it. Also, the search strategy he will adopt won't be case sensitive.

As you are very clever, Javinha asked you to write a program that automatically calculates the popularity of languages.

Input

The first line contains an integer P ($1 \leq P \leq 1000$), representing the number of posts in the forum archive. Each of the next P lines contains a post of at most 1000 characters (it can contain any ASCII character). The next line contains an integer N ($1 \leq N \leq 200$), representing the number of programming languages whose popularity should be calculated. Each of the next N lines contains the name of a programming language of at most 30 characters (it can contain any ASCII character).

Output

Output N lines, each one with the name of a programming language and the number of posts in which it was cited (separated by a single space). The languages must be printed in the same order they were informed in the input.

Examples

Input:	**Output:**
5 Is a Java object a pointer? I do not program in Java anymore. C, Java and Python are excellent languages. You forgot to free the memory. As a ruby programmer, I know how to use it. 6 JAVA C C++	JAVA 3 C 0 C++ 0 RUBY 1 PYTHON 1 SCALA 0

RUBY PYTHON SCALA	

Dica de solução (problema F):

Trata-se de um problema de busca em *strings*.

Como esta busca não é *case sensitive*, pode-se converter tanto o nome das linguagens quanto os *posts* para maiúsculas (ou minúsculas). Um ponto importante do enunciado diz respeito a uma linguagem ser contada em um *post* qualquer apenas quando o seu nome for precedido e sucedido por um espaço. Assim sendo, uma possível solução envolve: (1) Converter os nomes das linguagens e os *posts* para maiúsculas; (2) Para cada linguagem, procurar por seu nome precedido e sucedido de um espaço em cada post; (3) Para cada ocorrência, incrementar o contador de ocorrências da linguagem; (4) Exibir as linguagens e a quantidade de ocorrências na mesma ordem em que elas foram informadas na entrada.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema G

Gramáticas e linguagens

Arquivo fonte: `grama.{c | cpp | java}`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

A área de Linguagens Formais e Autômatos está no cerne da Teoria da Computação, e estuda as propriedades e conceitos que permitem determinar o que pode ser resolvido por meio de computadores e o que não pode. Um dos seus elementos fundamentais é o conceito de Gramática, um formalismo que indica como as palavras de uma linguagem são construídas. Conforme o tipo de gramática empregada será a complexidade das palavras que poderão ser formadas, e isso indica o tipo de linguagem com a qual estamos tratando. As linguagens mais simples são do chamado Tipo 3, as Linguagens Regulares; linguagens do Tipo 2, chamadas Linguagens Livres de Contexto, permitem construções mais complexas que as do Tipo 3; linguagens do Tipo 1, chamadas Linguagens Sensíveis ao Contexto permitem construções ainda mais sofisticadas e são apenas superadas nesse aspecto pelas linguagens do Tipo 0, as Linguagens Enumeráveis Recursivamente. O diagrama a seguir retrata a Hierarquia de Chomsky, que descreve o relacionamento entre esses tipos de linguagem. Perceba que as linguagens do Tipo 3 são um subconjunto das linguagens do Tipo 2 e estas, por sua vez, são parte das linguagens do Tipo 1 e assim por diante.

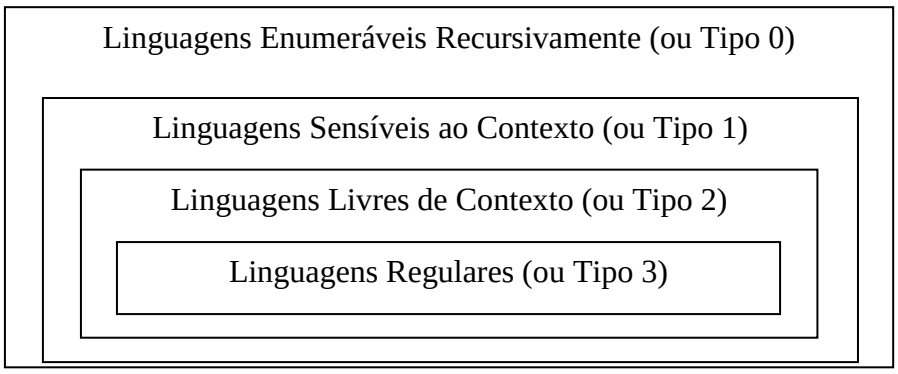


Fig 1: A hierarquia de Chomsky

Podemos identificar o tipo de uma gramática a partir da estrutura das regras utilizadas para formar as palavras. Uma regra tem sempre duas partes: o antecedente e o consequente. Ambos são strings que podem combinar três tipos de elementos: variáveis, terminais e o indicador de palavra vazia. Neste problema as variáveis serão sempre representadas por uma letra maiúscula, os terminais serão letras minúsculas e dígitos numéricos e o indicador de palavra vazia será o caracter '#'. Assim, por exemplo, o par (`aaXbc`, `aaa3`) é uma regra, onde o conteúdo antes da vírgula é o antecedente e o conteúdo após a vírgula é o consequente. Nesse exemplo encontramos no antecedente a variável *X* em meio a uma combinação dos terminais `a`, `b` e `c`, o consequente, por sua vez é composto apenas pela combinação dos terminais `aaa3`. Os parênteses não são considerados, pois servem apenas como delimitadores do par ordenado.

Para resolver este problema você não precisará entender como as regras são utilizadas para produzir as palavras de uma linguagem, precisa saber apenas que uma gramática precisa ter pelo menos uma regra embora geralmente possua várias. Além disso você precisará conhecer as características de cada tipo de gramática, que serão apresentadas resumidamente a seguir.

Se em todas as regras de uma gramática o antecedente é composto por um único caracter e esse caracter é uma variável; e além disso o consequente é formado sempre por um

único caracter, que é um terminal ou o indicador de palavra vazia ou então esse consequente é composto por dois caracteres sendo o primeiro um terminal e o segundo uma variável, essa é uma Gramática Linear à Direita e a linguagem por ela gerada é uma linguagem do Tipo 3.

Se todas as regras da gramática tiverem o antecedente composto por apenas um caracter, que seja uma variável e o consequente for composto pelo indicador de palavra vazia ou por alguma combinação qualquer de variáveis e terminais, essa é uma Gramática Livre de Contexto, e a linguagem que ela gera é uma linguagem do Tipo 2.

Uma gramática é chamada de Gramática Sensível ao Contexto se permite que o antecedente seja formado tanto por variáveis como por terminais (tem que ter pelo menos uma variável) e, além disso, o comprimento do antecedente é menor ou igual ao comprimento do consequente. Também é necessário que haja no máximo uma regra em que o consequente seja o indicador de palavra vazia; essa regra deve ser a primeira do conjunto de regras da gramática; o antecedente dela deve ser composto por um único caracter que é uma variável e essa variável não pode aparecer no consequente de nenhuma outra regra daquela gramática. Uma gramática com essas características permite gerar uma linguagem do Tipo 1.

Uma gramática válida que não se encaixe em nenhuma das situações anteriores é chamada de Gramática Irrestrita e gera linguagens do Tipo 0.

Entrada

O programa deverá processar vários casos de teste, onde cada caso é uma gramática a ser analisada. A primeira linha de um caso de teste é composta por um inteiro N ($1 \leq N \leq 20$) que indica a quantidade de regras da gramática. Seguem-se N linhas, cada uma com uma regra, no formato (A, C), onde A é uma string de até 20 caracteres indicando o antecedente da regra e C é uma string de até 20 caracteres indicando o seu consequente. Considere que apenas regras válidas serão fornecidas. Encerrar o processamento quando o programa ler um valor $N = 0$.

Saída

Para cada caso de teste, imprima "Tipo X" onde X é um número entre 0 e 3 que indica o tipo de linguagem que aquela gramática permite gerar. Atenção: você deve imprimir o tipo mais específico da linguagem gerada. Por exemplo, se uma gramática é Livre de Contexto, ela também se encaixa como sendo Sensível ao Contexto ou Enumerável Recursivamente (conforme podemos deduzir pela Hierarquia de Chomsky), mas seu programa deverá imprimir Tipo 2 (que é o mais específico nesse caso), e não Tipo 1 ou Tipo 0 na resposta.

Exemplos

Entrada:	**Saída:**
5 (P, aAbc) (A, aAbC) (A, #) (Cb, bC) (Cc, cc) 3 (S, aSbS) (S, bSaS) (S, #) 0	Tipo 0 Tipo 2

Dica de solução (problema G):

É um problema de processamento de strings. Receber cada regra da gramática e separar o antecedente do consequente, considerando que o antecedente é o que sobra da primeira palavra da linha ao descontar-se o parêntese inicial e a vírgula final; o consequente é a segunda palavra sem o parêntese final.

Em seguida, para cada regra, fazer a análise:

- Se o antecedente for composto apenas por uma variável (uma letra maiúscula), sabemos que o tipo será 2 ou 3. Para decidirmos qual dos dois tipos para aquela regra, verificamos se o consequente possui apenas um terminal (uma letra minúscula), ou um terminal com uma variável, ou apenas o símbolo '#'. Se uma dessas condições ocorrer, a regra é do tipo 3, senão é do tipo 2.
- Para as regras que não são do tipo 2 ou 3, precisamos analisar o consequente. Se ele atender aos requisitos para uma regra com '#' no consequente e garantia do antecedente mais curto que o consequente, a gramática será do tipo 1. Os casos restantes serão regras do tipo 0.

Com isso temos o tipo específico para cada regra da gramática. O tipo da gramática é o menor tipo dentre aqueles encontrados em suas regras.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

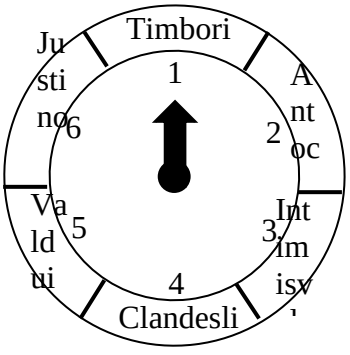
Operação Lava Ratos

Arquivo fonte: `roleta.{c | cpp | java}`

Autor: Julio Fernando Lieira (Fatec Lins)

A PI (Polícia Interplanetária) deflagrou inúmeras operações de busca e apreensão na investigação que ficou conhecida como Operação Lava Ratos, cuja quadrilha fazia uso de uma rede de Lavanderias e postos de combustíveis para lavar e abastecer naves intergalácticas clandestinas exterminadoras de pragas espaciais. Em uma destas operações foi apreendido um computador onde havia um sistema muito primitivo (o que indica que a corrupção vinha acontecendo a muito tempo) supostamente utilizado para distribuição do dinheiro conseguido de modo ilícito.

Master Yonenbaum, um auditor de TI já aposentado da PI, foi convocado para decifrar o funcionamento do sistema. Após dias de trabalho utilizando sua toolbox preferida (Minix, Turbo C, PC Tools Deluxe, DR DOS, etc), mestre Yonenbaum identificou que se tratava de um sistema que misturava a brincadeira da dança das cadeiras, relatada por seus avós, e uma espécie de jogo da roleta. Seu funcionamento se dava da seguinte maneira: sempre que havia uma grana a ser distribuída, os candidatos tinham seus nomes colocados em uma roleta, como na figura 1. A quantidade de casas da roleta é sempre igual a quantidade de participantes. O ponteiro da roleta sempre inicia no nome da casa 1 da roleta. Ao pressionar as teclas <Alt> <F9> o ponteiro da roleta gira no sentido horário avançando sempre o mesmo número de casas. O nome no qual o ponteiro para é então retirado da roleta. Por exemplo, a figura 2 mostra que o participante da casa 5 foi retirado após o ponteiro andar 4 casas e parar sobre seu nome. Após retirado o participante, os nomes dos demais participantes avançavam uma casa no sentido horário (como se a roleta com os nomes girasse 1 casa sentido horário), como mostra a figura 3. As teclas <Alt> <F9> eram pressionadas novamente e o ponteiro avançava o mesmo número de casas no sentido horário. O participante daquela casa era retirado e os nomes restantes avançavam uma casa. O último nome que restava era o feliz ganhador. Note que quando o ponteiro da roleta apontar para uma casa vazia, cujo participante já foi removido, conta-se a rodada, os participantes permanecem em suas posições e a roleta gira novamente.



Figura

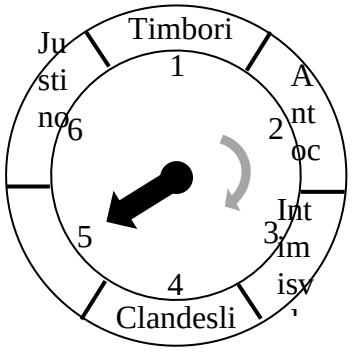
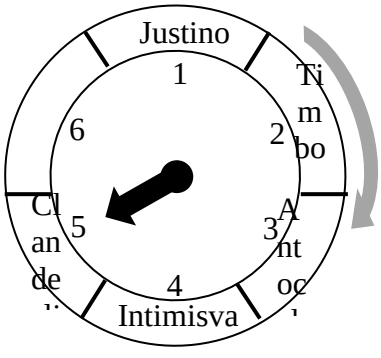


Figura 2



Figura

Como Mestre Yonenbaum está a muito tempo afastado de suas atividades, você, um jovem e promissor analista programador da Frota Interestelar, foi convocado para implementar o sistema e verificar se as informações do grande mestre conferem. Para tanto, algumas entradas e saídas extraídas de arquivos recuperados com o Undelete do PC Tools irão ajudar na sua missão.

Entrada

A entrada possui vários casos de teste. Cada caso inicia com um valor *N*, que indica a quantidade de participantes do sorteio. Na próxima linha, aparecem os *N* nomes dos participantes do sorteio, strings de até 20 caracteres alfanuméricos separadas por um caractere espaço. Na próxima linha segue um inteiro *P* que indica o número de casas que o ponteiro da roleta deve avançar a cada vez que forem pressionadas as teclas <Alt> <F9>. A entrada termina com o final do arquivo. Considere $2 \leq N \leq 100$; $1 \leq P \leq 1000$.

Saída

Para cada caso de teste, imprima a quantidade de rodadas e o vencedor. A quantidade de rodadas corresponde ao número de vezes que o ponteiro da roleta avançou, e o vencedor é o nome do participante que restou, depois que os demais foram eliminados. Cada palavra na saída deve ser separada por um espaço em branco e não se deve utilizar acentuação.

Exemplos

Entrada:

```
6
Timborio Antocelo Intimisval Clandeslio Valduino Justino
4
3
Sistencio YouSelf Cervejo
7
```

Saída:

```
apos 7 rodadas quem levou a bolada foi Clandeslio
apos 3 rodadas quem levou a bolada foi Sistencio
```

Dica de solução (problema H):

Este problema explora o conceito de estrutura circular, pois se a quantidade de casas que a roleta tem que avançar (P) é maior que a quantidade de nomes (N), avançar P é o mesmo que andar P mod N. Por exemplo, se N=6 participantes, andar 20 é o mesmo que andar 2 (20mod6).

Portanto, é preciso girar as duas roletas, primeiro a interna e depois a externa. Isso sugere um laço dentro de outro (laço externo avança o ponteiro da roleta de P em P (ou PmodN), e a cada rodada, avança a roleta externa com os nomes.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

InterFatocs. Você quis dizer InterFatecs?

Arquivo fonte: `sugestao.{c | cpp | java}`

Autor: Leandro Luque (Fatec Mogi das Cruzes)

Cezinho anda enfrentando problemas com a sua língua materna e também com a sua professora de português. Depois de ler em seus trabalhos frases como "Inglês não é meu forte, mas no português eu destróio" e "Eu odío isso", a professora de Cezinho ficou muito preocupada e resolveu investigar o que estava acontecendo. Ela descobriu que o menino usava e abusava das correções automáticas de alguns programas de computador e não se preocupava em escrever corretamente. O sistema de correção automática preferido dele é o do serviço de busca *web* que utiliza. Ele pesquisa "Sésar soro acaba" e o sistema corrige: "Você quis dizer César Sorocaba?", e por aí vai.

Procurando encaminhar o menino, a professora sugeriu que ele estudasse alguns materiais e também que fizesse um exercício relacionado à funcionalidade que ele gosta do sistema de busca. Para o exercício, a professora entregou a Cezinho duas listas, uma com palavras grafadas corretamente e outra com palavras grafadas incorretamente. A professora então escolhia uma palavra da lista de palavras com erro de grafia e pedia para Cezinho indicar qual palavra da lista de palavras grafadas corretamente ele iria sugerir. Ele tinha que escrever: "Você quis dizer <palavra sugerida da lista de palavras com grafia correta>".

Para encontrar uma sugestão de palavra, a professora o ensinou a calcular uma métrica de distância entre dois textos chamada de **Distância de Levenshtein**. Utilizando esta métrica, ele deveria então calcular a distância da palavra grafada incorretamente em relação a todas as palavras grafadas corretamente e escolher como sugestão aquela que tivesse a menor distância. Caso houvesse empate de distância entre duas ou mais palavras, a professora sugeriu que ele escolhesse a primeira destas palavras considerando a ordem alfabética.

A **Distância de Levenshtein** corresponde ao menor número de alterações (inserções, substituições ou deleções) que devem ser feitas em um texto *x* para se obter outro texto *y*. Uma forma de calculá-la (existem várias outras) é por meio de uma matriz como a apresentada a seguir, criada para calcular a distância entre as palavras VAZA e CASA.

		C	A	S	A
	0	1	2	3	4
V	1	1	2	3	4
A	2	2	1	2	3
Z	3	3	2	2	3
A	4	4	3	3	**2**

Para criar esta matriz, deve-se inicialmente preencher as células da primeira linha (1, j) com o valor do seu índice j e as células da primeira coluna (i, 1) com o valor do seu índice i. Em seguida, deve-se preencher o restante das células (i, j) da matriz baseando-se na seguinte regra

Matriz[i][j] = { Matriz[i-1][j-1], se *x*(i) = *y*(j)

Mínimo(Matriz[i-1, j] + 1, Matriz[i, j-1] + 1, Matriz[i-1, j-1] + 1), caso

contrário

onde *x*(i) representa o i-ésimo caractere do texto *x* e *y*(j) o j-ésimo caracetere do texto *y*.

Após calcular o valor para todas as células, o valor encontrado no canto inferior direito da tabela corresponde ao menor número de alterações que devem ser feitas em *x* para se chegar à *y*. No caso de VAZA e CASA, o menor número de alterações é igual a dois (2).

Apesar de conhecer muito bem esta versão do algoritmo da **Distância de Levenshstein**, Cezinho é muito preguiçoso e resolveu enganar sua professora. Ele pediu para o seu irmão, Javinha, um "cabra" muito esperto, para desenvolver um programa que, dada uma lista de palavras grafadas corretamente e uma palavra com grafia incorreta, faz uma sugestão conforme a regra definida pela professora de Cezinho. Como Javinha está empregado, ele pediu para você ajudá-lo na escrita deste programa.

Entrada

Inicialmente um valor N é informado, $1 \leq N \leq 100$, indicando a quantidade de palavras com grafia correta que serão informadas. Seguem-se N linhas, cada uma com uma palavra composta por até 20 caracteres maiúsculos sem espaço. Em seguida, é informado um valor I, $1 \leq I \leq 1000$, indicando o número de palavras com grafia incorreta que deverão ser verificadas. Seguem-se I linhas, cada uma com uma palavra composta por até 20 caracteres maiúsculos sem espaço.

Saída

Para cada palavra com grafia incorreta informada, imprima uma linha com a frase "voce quis dizer" (sem acento e em minúsculas), a palavra da lista de palavras grafadas corretamente que tem a menor distância de Levenshtein em relação à palavra com grafia incorreta e a distância de Levenshtein entre esta palavra e a palavra com grafia incorreta. Estas informações devem ser separadas por um espaço em branco. Caso, para uma determinada palavra com grafia incorreta, haja empate de distância entre duas ou mais palavras da lista de palavras grafadas corretamente, utilize a regra especificada para a professora para selecionar uma das palavras.

Exemplos

Entrada:	**Saída:**
6 MOGI SOROCABA SAOJOSE CRUZEIRO PINDA PIRAMBIRA 3 CRUSEIRO LOGI ROSOCABA	voce quis dizer CRUZEIRO 1 voce quis dizer MOGI 1 voce quis dizer SOROCABA 2

Dica de solução (problema I):

A solução deste problema envolve a implementação do algoritmo da Distância de Levenshtein descrito no enunciado (ele não será descrito novamente). Para cada palavra com

erro de grafia informada, deve-se calcular a distância dela para todas as palavras com grafia correta. Em seguida, deve-se encontrar a menor destas distâncias e, no caso de existirem duas distâncias iguais, a primeira destas palavras considerando a ordem alfabética deve ser escolhida. Portanto, além do algoritmo da distância, deve-se usar um algoritmo de ordenação já disponível na linguagem ou implementar um.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J

O problema do aniversário

Arquivo fonte: `niver.{c | cpp | java}`

Autor: Antonio Cesar de Barros Munari (Fatec Sorocaba)

Malaquias está intrigado com um assunto que viu na aula de Matemática. Era o chamado Paradoxo (ou problema) do Aniversário: seu professor falou que em um grupo de 23 pessoas, a probabilidade de que duas delas façam aniversário no mesmo dia é de aproximadamente 50%, um número que Malaquias considerou surpreendentemente alto. E se forem 32 pessoas, a chance de duas dessas pessoas aniversariarem na mesma data passa de 60%! Realmente, temos que reconhecer que é algo um pouco contra intuitivo. Então Malaquias deseja colocar à prova essa afirmação, analisando para o conjunto de alunos de sua classe as datas de nascimento de todos e verificando se encontrará pessoas aniversariando no mesmo dia que ele. E você vai ajuda-lo com um programinha de computador.

Entrada

O seu programa será testado com vários casos de teste. Inicialmente dois inteiros positivos D e M são informados, separados por um espaço em branco, indicando, respectivamente, o dia e o mês em que Malaquias nasceu, com $1 \leq D \leq 31$ e $1 \leq M \leq 12$. Caso seja informado um valor $D = 0$ junto com um valor $M = 0$, o programa deverá encerrar o processamento. Em seguida um valor inteiro positivo Q é informado, $1 \leq Q \leq 100$, indicando quantos colegas Malaquias tem em sua classe. Seguem-se Q linhas, cada uma contendo um par de inteiros X e Y, representando o dia e o mês de nascimento de cada membro da turma, com $1 \leq X \leq 31$ e $1 \leq Y \leq 12$.

Saída

Para cada caso de teste, imprima a letra S (em maiúscula) se foi encontrada na turma pelo menos uma pessoa que faça aniversário no mesmo dia e mês que Malaquias. Se ninguém da turma faz aniversário junto com ele, imprima uma letra N (em maiúscula). Ao imprimir o resultado não se esqueça de finaliza-lo com uma quebra de linha.

Exemplos

Entrada:	**Saída:**
25 3 4 10 4 5 1 25 3 31 1 12 8 10 5 11 11 8 13 8 22 2 1 1 25 12 10 4 11 8 20 5	S N

6 10 0 0	

Dica de solução (problema J):

Problema mais fácil da prova, para cada caso de teste bastava receber a data de aniversário inicial e depois verificar, para cada data fornecida na sequência, se esta era igual à data inicialmente informada. Como era necessário receber todas as datas do caso, a solução era manter um flag para indicar se alguma data coincidente foi encontrada. No início da entrada de dados de um caso coloca-se um valor Falso para o flag, ao encontrar uma data igual à data inicial muda-se o valor do flag para Verdadeiro e continua-se a receber e processar as novas datas. Ao final, caso o flag esteja com valor Verdadeiro, imprime-se ‘S’, senão imprime-se ‘N’.